UNIVERSITÉ DE MONTRÉAL

CENTRE DE RECHERCHE SUR LES TRANSPORTS

CASE POSTALE 6128, SUCCURSALE A, MONTRÉAL, QUÉBEC, H3C 3J7

OPTIMAL ROUTING UNDER CAPACITY AND DISTANCE RESTRICTIONS

by

Gilbert Laporte*, Yves Nobert**
and Martin Desrochers*

The authors are grateful to the Canadian Natural Sciences and Engineering Research Council (grants A4747 and A5486) and to the Québec Government (F.C.A.C. grant 80EQ0428) for their financial support.

* Ecole des Hautes Etudes Commerciales de Montréal
5255, av. Decelles, Montréal H3T 1V6

** Département des Sciences administratives, Université du Québec à Montréal, 1495, rue Saint-Denis, Montréal H3C 3P8

Université de Montréal

Centre de recherche sur les transports - Publication #328
Ecole des Hautes Etudes Commerciales -
Les Cahiers du Gérad - G-83-18

Août 1983

Dépôt légal - 1er trimestre 1984

ABSTRACT

This paper describes an integer linear programming algorithm for vehicle routing problems involving capacity and distance constraints. The method uses constraint relaxation and a new class of subtour elimination constraints. Two versions of the algorithm are presented, according to the nature of the distance matrix. Exact solutions are obtained for problems involving up to 55 cities.

RESUME

Cet article présente un algorithme de programmation linéaire en nombres entiers pour le problème de tournées de véhicules avec contraintes de capacité et de distance. La méthode fait appel à la relaxation de contraintes et à une nouvelle classe de contraintes d'élimination de sous-tours. Deux versions de l'algorithme sont présentées, dépendant de la nature de la matrice de distances. L'algorithme permet d'obtenir la solution exacte de problèmes contenant jusqu'à 55 villes.

1. INTRODUCTION

The *vehicle routing problem* (VRP) is usually defined as the problem of optimally designing delivery routes for a set of vehicles. The operational research literature contains several versions of the VRP and a number of algorithms for their solution (see [1,3,6] for excellent surveys). This paper describes the construction of an exact algorithm for the solution of VRP's having the following characteristics.

(i) We consider a symmetrical graph $G = (N,E)$ where $N = \{1,\ldots,n\}$ is a set of nodes (cities, customers) and $E = \{(i,j): i,j \in N, i<j\}$, a set of undirected edges. $C = (c_{ij})$ is the matrix of distances associated with the edges. (In this paper, c_{ij} must be interpreted as c_{ji} whenever $i > j$.) C is said to be *Euclidean* if and only if $c_{ij} \leq c_{ik} + c_{kj}$ $(i,j,k \in N)$. N includes a *depot* located at city 1. With each city i is associated a nonnegative weight or requirement d_i; by convention, $d_1 = 0$.

(ii) Identical vehicles having the same capacity D are based at the depot $(D \geq \max \{d_i\})$.

(iii) Each city, except the depot, must be visited by only one vehicle exactly once.

(iv) Each vehicle starts and ends its journey at the depot. The sum of weights contained on a vehicle's route may not exceed D and the total length of the route may not exceed a prespecified upper bound L.

(v) The objective is to minimize the total distance travelled while satisfying all constraints.

In spite of an abundant literature on the (symmetrical) VRP, few exact algorithms exist for its solution. The work which appears to be the most relevant to this paper is by Christofides [2], Christofides, Mingozzi and Toth [4] and Laporte, Nobert and Desrochers [12,13].

The algorithm developed by Christofides [2] is based on the generation of all feasible routes by means of an implicit enumeration tree. The largest problem reported solved by the author involved 31 cities; this problem contained both capacity and distance constraints. Christofides et al. [4] later suggested two branch and bound algorithms. In the first one, the bounds are derived from m-trees [8] and in the second one, from q-paths (i.e. chains of nodes whose total weight is equal to q). The second algorithm appears to be superior on the whole. It has been tested on problems ranging from 10 to 25 cities and involving capacity constraints only.

The papers by Laporte and Nobert [12] and by Laporte et al. [13] consider problems with capacity and distance constraints respectively. The VRP is formulated as an integer program which is solved by means of a constraint relaxation procedure. Integrality is reached by branch and bound in both papers and also by a cutting planes method in the second one. Computational results are reported for problems ranging from 15 to 50 cities in the case of the capacity constrained VRP and from 20 to 60 cities in the other case.

This paper describes an exact algorithm for VRP's involving capacity and distance constraints simultaneously. It extends the work initiated in [12,13] and takes advantage of structural similarities between the two problems. As we shall see, Euclidean and non-Euclidean problems possess quite different properties and will therefore be treated separately.

2. NOTATION

Apart from the notation already introduced above, we define

(i) m: the number of vehicles in the optimal solution. m is either *fixed* if its value is determined a priori or *free* if its value is determined simultaneously with the best set of routes. Our algorithm applies to both cases.

(ii) $\bar{S} = N-S \qquad (S \subseteq N)$

(iii) $S^+ = S \cup \{1\}$ and $S^- = S - \{1\} \qquad (S \subseteq N)$

(iv) $\lceil t \rceil = \begin{cases} \text{the smallest integer greater than or equal to } t \text{ if } t > 0 \\ 1 \text{ if } t \leq 0 \end{cases}$

(v) $H_v(S) = \begin{cases} \text{the optimal value of the multiple travelling salesman problem [11] solution through all nodes of S (involving no node outside S and where v is the fixed number of salesmen) if } 1 \in S \text{ and } S \subseteq N \\ \text{the optimal value of the TSP solution through all nodes of S (and involving no node outside S) if } v = 1 \\ \text{undefined otherwise} \end{cases}$

(vi) V(S): a lower bound on the number of vehicles required to visit all nodes of S^+ in the optimal solution to the VRP on N

(vii) In a branch and bound algorithm (see section 4), the superscript h (e.g. in $H_v^h(S)$, $V^h(S)$, etc.) restricts the definition of the variable to node h of the search tree.

(viii) $x_{ij} = \begin{cases} 1 & \text{if a vehicle is used on a single trip between i and j } (i,j \in N) \\ 2 & \text{if a vehicle is used on a return trip between i = 1 and j} \\ 0 & \text{otherwise} \end{cases}$

N.B. x_{ij} is only defined if $i < j$, $d_i + d_j \leq D$ and $c_{ij} \leq L$. x_{ij} must be interpreted as x_{ji} whenever $i > j$.

3. EUCLIDEAN PROBLEMS: FORMULATION

Euclidean problems are formulated as follows.

(P1) minimize $z = \sum_{i,j \in N} c_{ij} x_{ij}$

subject to

(1) $\sum_{j \in N^-} x_{1j} = 2m$

(2) $\sum_{i<k} x_{ik} + \sum_{j>k} x_{kj} = 2$ $\qquad (k \in N^-)$

(3) $\sum_{i,j \in S} x_{ij} \leq |S| - V(S)$ $\qquad (S \subseteq N^-,\ |S| \geq 3)$

(4) $x_{ij} = \begin{cases} 0, 1, 2 & (i=1,\ j \in N^-,\ c_{1j} \leq \frac{1}{2}L) \\ 0, 1 & \text{otherwise} \end{cases}$

$m \geq 1$ and integer

In this formulation, m can either be a constant (m fixed) or a variable (m free). Constraints (1) and (2) specify the degree of each node while constraints (4) deal with the integer requirements on the variables. Constraints (3) are *subtour elimination constraints*: they prohibit the occurence of illegal subtours, i.e. subtours which are either

(i) disconnected from the depot, or

(ii) connected to the depot and having a total weight exceeding D, or

(iii) connected to the depot and having a total length exceeding L.

Their form is the same as that used for the TSP in Dantzig et al. [5] and in subsequent papers; but the right-hand side is equal to $|S| - r$, where $r \geq 1$, instead of $|S| - 1$. The constraints are therefore stronger than those used by Dantzig et al. The computation of V(S) will be explained in the next section as it is not known a priori. It is easy to show [14] that the following equivalence (proved for $r = 1$ in [5]) is also valid as long as the degree of each node of S is equal to 2:

$$(5) \quad \sum_{i,j \in S} x_{ij} \leq |S| - r \quad \Leftrightarrow \quad \sum_{\substack{i \in S, j \in \bar{S} \\ \text{or } i \in \bar{S}, j \in S}} x_{ij} \geq 2r$$

From a computational point of view, it is preferable to use the form involving the least number of coefficients.

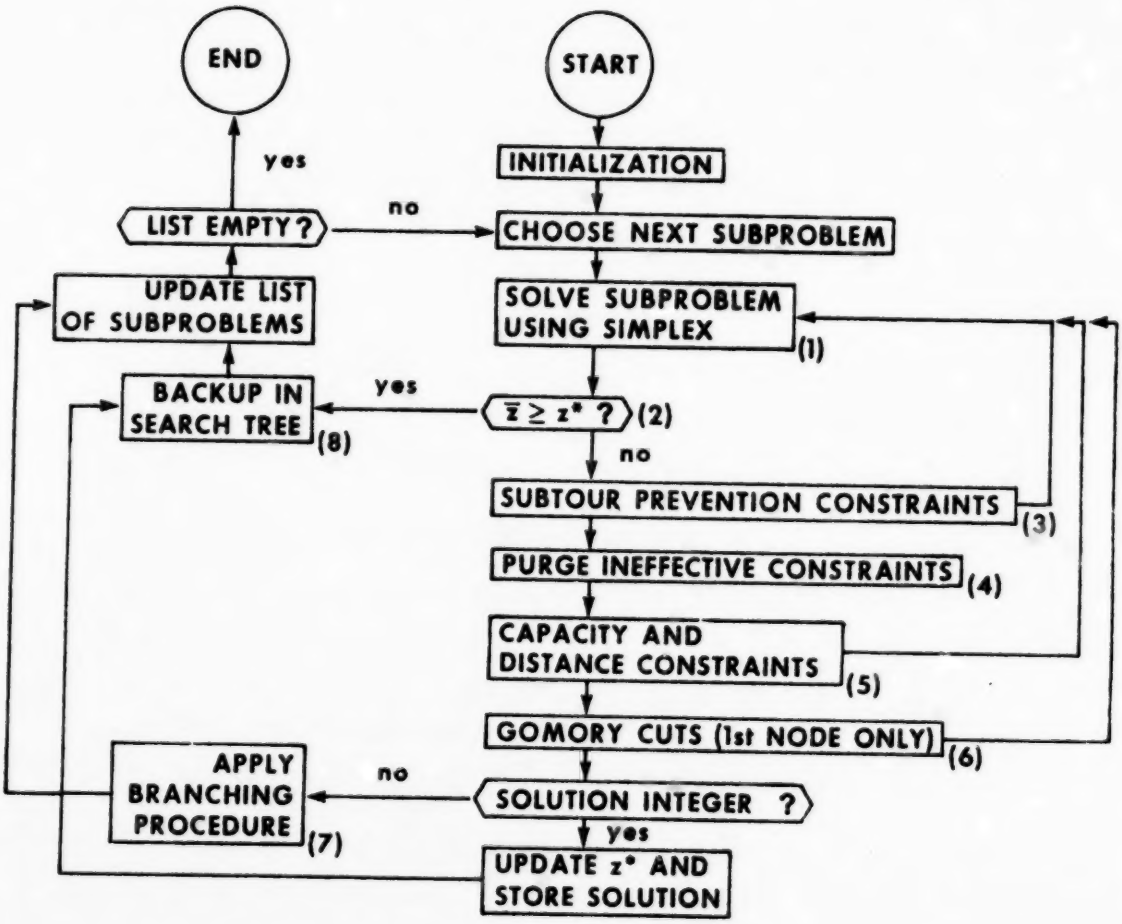
4. EUCLIDEAN PROBLEMS: ALGORITHM

As in [12,13], we solve (P1) by first relaxing the integrality and subtour elimination constraints, and by generating them only when they are discovered to be violated. Preliminary tests have indicated that (P1) could not be solved by a pure cutting planes method (as was the case in [13] for example) and that a branch and bound approach would have to be used to reach integrality. The order in which the various types of infeasibilities are investigated and then eliminated is quite critical (and in fact is not the same in the case of Euclidean and non-Euclidean problems). Only the order yielding the best computational performance is presented in this paper.

The algorithm can be summarized by the flow chart presented in figure 1. To a large extent, this diagram requires little explanation in so far as it describes a standard branch and bound procedure to solve integer programs. Interested readers are refer-

FIGURE 1

Flow chart of the algorithm for Euclidean problems



red to [9] for details. Steps requiring additional comments have been numbered and are described in greater length.

Step 1: Solve subproblem using simplex

We use here the Land-Powell code (LINP subroutine [9]) to solve linear programs. The main characteristic of this code is its use of a reduced working inverse basis whose size varies throughout the algorithm and is in fact equal to the number of effective constraints at a given iteration. This produces substantial savings in computational time and memory requirements. LINP is also very appropriate to our type of approach as it enables ineffective constraints to be deleted and new (violated) ones to be generated quite easily.

Step 2: $\bar{z} \geq z^*$?

z^* = the cost of the best solution so far identified

$\bar{z}$ = the cost of the current subproblem ($\bar{z} = \infty$ whenever the subproblem is infeasible).

Step 3: Subtour prevention constraints

Consider the solution at a given node h of the search tree: this solution contains

(i) sets of nodes $\{i_1,\ldots,i_u\}$ $(u > 1)$ corresponding to chains $(i_1,\ldots,i_u)$ such that $1 \notin \{i_2,\ldots,i_{u-1}\}$ if $u > 2$ and for which *all* variables $x_{i_1 i_2}, x_{i_2 i_3}, \ldots, x_{i_{u-1} i_u}$ have been *fixed* at 1, and

(ii) nodes not belonging to such chains (we define for each such node i a set $\{i\}$).

For convenience, we refer to these sets of nodes S_k (corresponding to chains or single nodes) as *components*. Each S_k has an associated weight $w(S_k)$ defined as

$$(6)\quad w(S_k) = \sum_{i \in S_k} d_i$$

and a length $\ell^h(S_k)$ defined as

$$(7)\quad \ell^h(S_k) = \begin{cases} \sum_{t=1}^{u-1} c_{i_t i_{t+1}} & \text{in the case of a chain } (i_1,\ldots,i_u) \\ 0 & \text{in the case of a node} \end{cases}$$

Consider a component S_r. If it corresponds to a chain, let p_r and q_r be the end nodes of that chain; if it corresponds to a node i, let $p_r = q_r = i$. In the first case, variable $x_{p_r q_r}$ can be *forced* to zero if

(i) $p_r, q_r \in N^-$ or

(ii) $p_r = 1$ and $\ell^h(S_r) + c_{1 q_r} > L$.

Now consider two components S_r and S_s and let $i \in \{p_r, q_r\}$, $j \in \{p_s, q_s\}$. Variable x_{ij} can be forced to zero if

(iii) $i,j \in N^-$ and $w(S_r) + w(S_s) > D$, or

(iv) $i,j \in N^-$ and $\ell^h(S_r) + \ell^h(S_s) + c_{ij} > L$, or

(v) $i=1$, $j \neq 1$ and $\ell^h(S_s) + c_{1j} > L$.

In practice, forcing variables to zero is done by introducing into the current subproblem a *subtour prevention constraint* stating that the sum of all such variables cannot exceed zero. This constraint is only introduced explicitly in the program when it becomes violated. The effects of this constraint are to prevent the occurence of illegal subtours rather than eliminating them after they have been identified

and to increase the value of the objective function at an earlier stage in the search tree.

Step 4: Purge ineffective constraints

Constraints (other than degree constraints) generated at a higher level or at the same level in the search tree and which are currently ineffective are eliminated from the program. In some rare instances, these constraints may have to be generated again, but in general, this practice has proved a convenient time and memory saving device. Degree constraints are not removed as they tend to be effective in the optimal solution. This operation is carried out by the subroutine PURGE developed by Land and Powell [9] and adapted to the VRP by Nobert [14].

Step 5: Capacity and distance constraints

At node h of the search tree, consider the graph $G' = (N^-, E')$ where E' is the set of all (undirected) edges (i,j) for which $i,j \in N^-$ and $x_{ij} > 0$. N^- can be partitioned into a number of *connected components* $S_1, S_2, \ldots$, i.e. sets of nodes linked by a chain of "positive edges". Each S_k^+ has a weight

$$(8) \quad w(S_k^+) = w(S_k) = \sum_{i \in S_k} d_i$$

and a length

$$(9) \quad \ell^h(S_k^+) = \sum_{i,j \in S_k^+} c_{ij} x_{ij}$$

It can be shown that the minimum number of vehicles required to visit all nodes of S_k^+ in the optimal solution is bounded below by $\lceil w(S_k)/D \rceil$ and by the smallest value of v satisfying

(10) $v = \lceil H_v^h(S_k^+)/L \rceil \qquad (v = \lceil \ell^h(S_k^+)/L \rceil, \ldots, |S_k|)$

In practice, this value is easily determined: tests have shown that (10) is generally satisfied for $v = \lceil \ell^h(S_k^+)/L \rceil$ which is clearly a valid lower bound.

Therefore, for each k, we can set

(11) $V^h(S_k) = \max \{\lceil w(S_k)/D \rceil, \lceil \ell^h(S_k^+)/L \rceil\}$

and impose the following subtour elimination constraint, whenever it is violated.

(12) $\sum_{i,j \in S_k} x_{ij} \leq |S_k| - V^h(S_k)$

(Note: if h' is a descendant of node h, $V^{h'}(S_k) \geq V^h(S_k)$ and constraint (12) remains valid for h'.)

Step 6: Gomory cuts (1st node only)

Tests have shown [14] that generating Gomory cuts [7,9] at the first node of the search tree increases the initial lower bound on the problem and helps to obtain an integer solution faster. We may generate up to 5 Gomory cuts as in general, after that number, they tend to be less efficient. No cuts were introduced at higher levels in the search tree as their generation and their elimination when backtracking (step 8) requires substantial computational effort.

Step 7: Apply branching procedure

We use the Land-Powell [9] branching procedure: the (fractional) variable chosen is the one which would produce the largest increase in the objective function if it was branched upon in a given direction, but branching is first made in the opposite direction. We have also experimented with a second branching procedure, namely, selecting the

variable which would produce the smallest increase in the objective function if it was fixed at 1 and branching in that direction. The rationale of this rule is to produce as quickly as possible chains of variables containing variables fixed at 1 and thus increase the likelihood that subtour prevention constraints will be generated. Tests indicate that the second strategy yields more subtour prevention constraints and produces a feasible solution sooner. On the average, it is however slower than the first strategy and has not been retained.

Step 8: Backup in search tree

Here we use the Land-Powell BACKUP subroutine [9] with the following modification. When backtracking in the search tree from level λ' to level $\lambda(\lambda' \geq \lambda)$, it is essential to remove all subtour prevention constraints and to modify all subtour elimination constraints whose validity depends on which variables were fixed at levels $\lambda, \lambda+1, \ldots, \lambda'$ of the search tree. Subtour prevention constraints clearly depend on which variables are fixed at 1 at these levels. Subtour elimination constraints fall into two categories (see (11) and (12)):

(i) those which were generated with $V^h(S_k) = \lceil w(S_k)/D \rceil$

(ii) and those which were generated with $V^h(S_k) = \lceil \ell^h(S_k^+)/L \rceil > \lceil w(S_k)/D \rceil$

The first ones are valid throughout the algorithm and need not be eliminated since the value of $w(S_k)$ does not depend on h. In the second case, the value of $\ell^h(S_k^+)$ depends on which variables were fixed at 0, 1 or 2 at node h. Therefore, it is necessary to increase the right-hand side of such constraints from $|S_k| - \lceil \ell^h(S_k^+)/L \rceil$ to $|S_k| - \lceil w(S_k)/D \rceil$.

Removing a subtour prevention constraint of the form $ax \leq b$ requires first replacing b by a suitably large number; modifying a subtour elimination constraint also requires increasing its right-hand side. In both cases, the current solution has to be updated in order to

take these changes into account (see [10] for technical details), simplex iterations have to be executed and all constraints made ineffective are "purged" [9].

5. NON-EUCLIDEAN PROBLEMS: FORMULATION

When C is non-Euclidean, some of the results contained in step 5 of the previous section do not hold. In particular, the value of $\lceil e^h(S_k^+)/L \rceil$ is no longer a valid lower bound on the number of vehicles required to visit all nodes of S_k^+ in the optimal solution. Equation (11) and constraint (12) no longer hold. However, it is still correct to set the value of $V^h(S_k)$ at $\lceil w(S_k)/D \rceil$. A new type of constraint is required to eliminate connected components satisfying capacity constraints, but not distance constraints.

The formulation used for non-Euclidean problems is similar to (P1) except that constraints (3) are replaced by

$$(13) \quad \sum_{i,j \in S} x_{ij} \leq |S| - \lceil w(S_k)/D \rceil \qquad (S \subseteq N^-,\ |S| \geq 3)$$

and

$$(14) \quad \sum_{j \in S^-} x_{1j} + 3 \sum_{\substack{i \in S^-, j \in \bar{S} \\ \text{or } i \in \bar{S}, j \in S^-}} x_{ij} \geq 4 \qquad (S \subseteq N,\ 1 \in S,\ |S| \geq 3,\ H_1(S) > L)$$

The validity of constraints (13) has already been established in [12]: they clearly eliminate any subtour disconnected from the depot or connected to the depot, but having a total weight exceeding D. Constraints (14) require a more detailed explanation.

Consider the solution of the subproblem at node h of the search tree and assume this solution contains a subtour involving nodes of a set S such that

(i) $1 \in S$,

(ii) $\ell^h(S) = \sum_{i,j \in S} c_{ij} x_{ij} > L$ and

(iii) $w(S) = \sum_{i \in S} d_i \leq D$.

Constraints (14) are obtained as follows. Since the subtour is illegal, neither it, nor any permutation of its nodes can appear in the optimal solution (since this would clearly yield a subtour with a length larger than L). Therefore, it can be eliminated

(i) either by *partitioning* S into several subsets, each subset corresponding to a subtour connected to the depot:

(15) $$\sum_{j \in S^-} x_{1j} \geq 4$$

(ii) or by creating *exactly one new link* between S^- and $\bar{S}$; in this case, there will have to be at least one link between S^- and the depot:

(16) $$\sum_{j \in S^-} x_{1j} \geq 1 \quad \text{and} \quad \sum_{\substack{i \in S^-, j \in \bar{S} \\ \text{or } i \in \bar{S}, j \in S^-}} x_{ij} = 1$$

(iii) or by creating *at least two new links* between S^- and $\bar{S}$:

(17) $$\sum_{\substack{i \in S^-, j \in \bar{S} \\ \text{or } i \in \bar{S}, j \in S^-}} x_{ij} \geq 2$$

We seek a constraint which eliminates the current (infeasible) solution characterized by

$$(18)\quad \sum_{j\in S^-} x_{1j} = 2 \quad \text{and} \quad \sum_{\substack{i\in S^-, j\in \bar{S} \\ \text{or } i\in \bar{S}, j\in S^-}} x_{ij} = 0$$

It can be seen from figure 2 that the two constraints defined by (19) and (20) eliminate this infeasibility without cutting the feasible domain:

$$(19)\quad \sum_{j\in S^-} x_{1j} + \sum_{\substack{i\in S^-, j\in \bar{S} \\ \text{or } i\in \bar{S}, j\in S^-}} x_{ij} \geq 2$$

and

$$(20)\quad \sum_{j\in S^-} x_{1j} + 3 \sum_{\substack{i\in S^-, j\in \bar{S} \\ \text{or } i\in \bar{S}, j\in S^-}} x_{ij} \geq 4$$

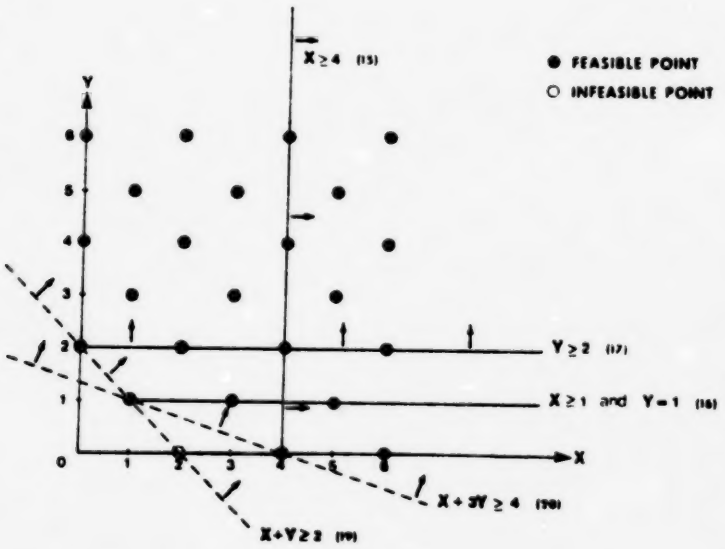
It is straightforward to prove that (19) is equivalent to

$$(21)\quad \sum_{i,j\in S^-} x_{ij} \leq |S^-| - 1$$

i.e. the standard TSP subtour elimination constraint [5]. But (21) is dominated by (13) and therefore (19) need not be included in the formulation. It can also be shown that (20) is dominated by (13) whenever $\lceil w(S_k)/D \rceil \geq 2$.

FIGURE 2

Subtour elimination constraints for non-Euclidean problems



$$X = \sum_{j \in S^-} x_{1j} \quad \text{and} \quad Y = \sum_{\substack{i \in S^-, j \in \bar{S} \\ \text{or } i \in \bar{S}, j \in S^-}} x_{ij}$$

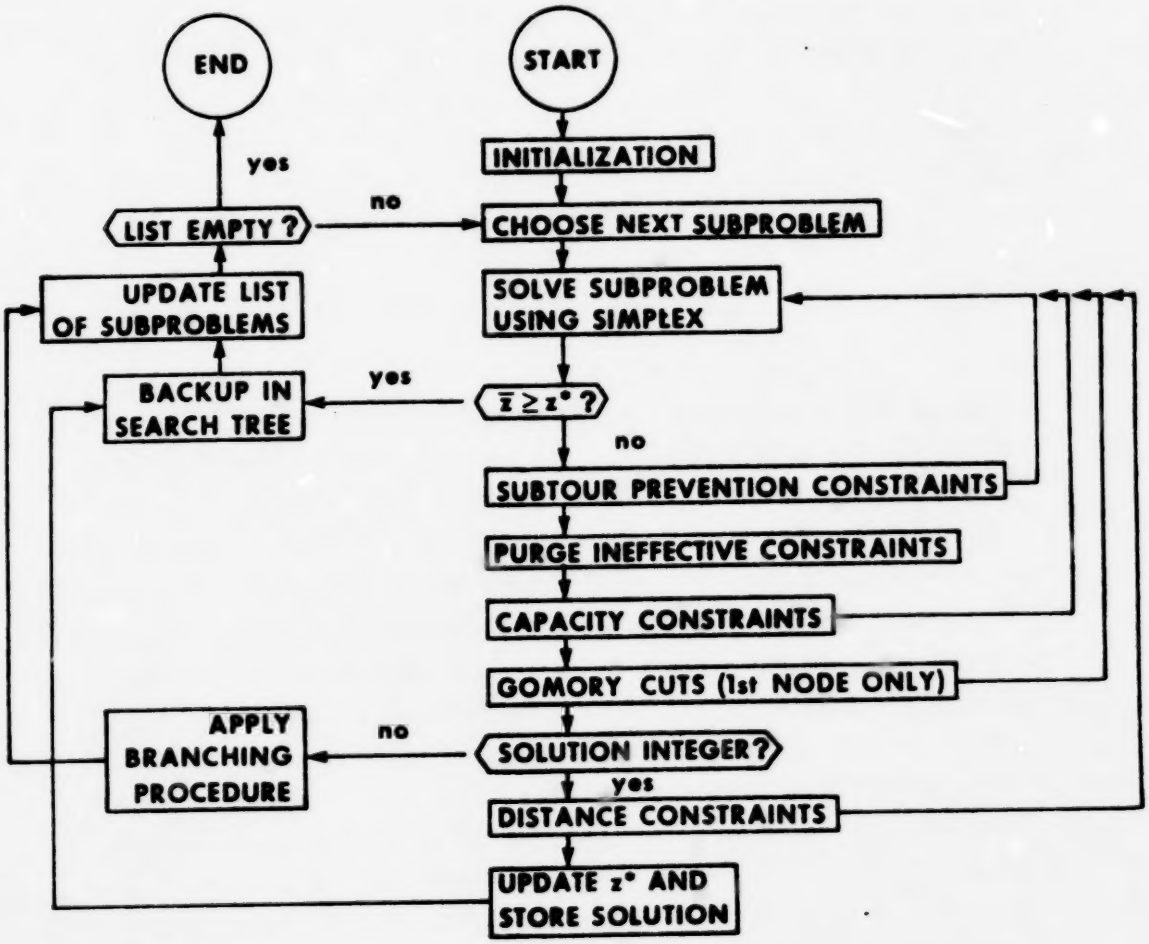
6. NON-EUCLIDEAN PROBLEMS: ALGORITHM

We use the same type of algorithm for non-Euclidean problems and for Euclidean problems. However, as mentioned earlier, some steps are performed in a different order. Figure 3 illustrates the flow chart of the modified algorithm.

Note that capacity constraints (defined by (13)) and distance constraints (defined by (14)) are generated separately and

FIGURE 3

Flow chart of the algorithm for non-Euclidean problems



that violated distance constraints are only introduced after an integer solution has been obtained. These must be removed when backtracking (see section 4, step 8) by first replacing their right-hand sides by -1 and then by applying the procedure described in [10].

7. COMPUTATIONAL RESULTS

The algorithms presented in sections 4 and 6 were tested on a number of problems randomly generated as follows. For Euclidean problems, n points (x_i,y_i) were first uniformly generated on $[0,100]^2$ and the c_{ij}'s were defined as $c_{ij} = [(x_i-x_j)^2 + (y_i-y_j)^2]^{\frac{1}{2}}$. The d_i's and the c_{ij}'s for non-Euclidean problems were obtained from a uniform distribution on [0,100]. The vehicle capacity D was determined through the following formula:

$$(22)\quad D = (1 - \alpha) \max_{i=1,\ldots,n} \{d_i\} + \alpha \sum_{i=1}^{n} d_i$$

where α is a parameter chosen in [0,1]. Each problem was successively solved with four values of α: 0, .33, .67 and 1. A low value of α yields a tight problem; when $\alpha = 1$ and when there are no distance constraints, the problem reduces to a multiple TSP [11]. For each problem, four values of L were also chosen between 150 and 1000. In problems with L = 1000, the distance constraint was not binding. Problems with low values of α and L are the most difficult to solve with our algorithms because of the large number of subtour elimination constraints which have to be generated; on the other hand, algorithms based on the implicit enumeration of feasible solutions [2,4] perform relatively well on these tight problems since fewer alternatives have to be (implicitly) explored. The reverse is also true. In all test problems, m was free; previous experience [11,12] with the type of algorithm described in this paper has shown that problems with m free are in general more difficult to solve than those with m fixed.

All problems were run on the University of Montreal CYBER 173 computer, using an FTN5 compiler. The maximum CPU time allowed in each case was set at 500 seconds. (Problems "failed" when they could

not be solved within that time.) The memory available was $200\ 000_8$ words, but in no case was this limit reached.

Average results for the number of successful problems (out of 3) are presented in tables 1 and 2. The column labelled "maximum size of the inverse" (see [9]) represents the maximum number of effective constraints at any stage in the branch and bound tree. In the case of Euclidean problems, the subtour elimination constraints are defined by (11) and (12); in the case of non-Euclidean problems capacity constraints are defined by (13) and distance constraints by (14).

The largest problems which were solved contained 35 cities in the case of Euclidean problems and 55 cities in the case of non-Euclidean problems. For larger values of n, no problems with $\alpha = 0$ or .33 could be solved within 500 seconds. Results for problems involving capacity restrictions only (i.e. L = 1000) or distance restrictions only (i.e. $\alpha = 1$) are comparable to those found in [12] and in [13]. As expected, problems with small values of α and L required more subtour elimination constraints and were more difficult to solve.

Relatively few constraints were generated, considering their potential number. In general, subtour prevention constraints could only be imposed in fairly tight problems and in those for which n was small. Many problems (especially those with $\alpha = .67$ or 1) required little or no branching; however, as mentioned earlier, a pure cutting planes approach proved infeasible in many situations, because of the unacceptable increase in the size of the determinant of the inverse basis.

TABLE 1

Results for Euclidean problems

n	π	L	number of successful problems out of 3	time (seconds)	number of simplex iterations	maximum size of the inverse	maximum level of the search tree	number of Gomory cuts	number of subtour elimination constraints	number of subtour prevention constraints
15	0	150	2	8.50	384	28.5	5.5	3.5	42.5	4
		175	3	19.76	414	23.3	7	4	26.1	1.3
		200	3	48.11	932.7	28.7	10	4	57	7.7
		1000	3	47.93	932.7	28.7	10	4	57	7.7
	.33	150	3	93.3	1535.3	30.3	9	5	169.7	1
		175	2	103.74	1821.5	28	14	5	93.5	0
		200	3	89.45	1801.3	29.7	17	5	84	0
		1000	3	52.64	1137.3	28.7	16.7	5	44	0
	.67	150	2	98.93	1535	29	9	5	174.5	.5
		175	3	86.7	1707.7	25.3	14.3	3.7	123.7	2.3
		200	3	16.59	419	22.3	3.3	2	33.4	0
		1000	3	1.74	51	22	.3	2.3	6.3	0
	1	150	3	201.71	3323	37.7	11	3.3	251	4.7
		175	3	17.79	382	26.3	5	3.7	37.3	0
		200	3	30.82	654.3	23.3	3.3	2.3	51	0
		1000	3	1.1	47.7	18.3	0	1.3	2.3	0
20	0	175	0	-	-	-	-	-	-	-
		200	1	295.19	2864	43	25	5	154	15
		225	1	360.91	3714	44	24	5	164	20
		1000	1	361.12	3714	44	24	5	164	20
	.33	175	2	180.36	2355.5	35.5	11.5	5	161	0
		200	2	45.79	577.5	32	8	5	30.5	0
		225	3	31.43	410	30.3	5.3	3.3	23.6	0
		1000	3	31.52	410	30.3	5.3	3.3	23.6	0
	.67	175	3	173.01	2379.7	34.3	13	5	151.7	0
		200	3	14.12	292.3	30	1.3	2.7	26	0
		225	3	9.66	139	29	2	3.3	19.3	1.3
		1000	3	15.13	243	27	3	1.7	14.3	0
	1	175	3	101.92	1106.3	34.7	20.3	5	65.7	.7
		200	3	11.96	149.7	30.7	2	3.3	23.3	0
		225	3	15.52	101.1	29.7	3.3	5.3	9	0
		1000	3	3.17	61	26.7	0	2.6	7.3	0
25	.33	200	0	-	-	-	-	-	-	-
		225	1	51.58	397	41	8	5	32	0
		250	1	51.43	397	41	8	5	32	0
		1000	1	81.81	841	42	11	5	33	0
	.67	200	0	-	-	-	-	-	-	-
		225	0	-	-	-	-	-	-	-
		250	1	79.42	1011	38	13	5	25	0
		1000	3	50.9	621	38.7	5.3	3.7	14	0
	1	200	1	17.64	197	40	0	4	25	0
		225	3	141.21	1336.3	39.7	25.3	3.3	67.4	0
		250	1	218.05	1559	42	18	5	122	0
		1000	3	6.88	99	33	0	2.3	9.7	0
30	.33	200	0	-	-	-	-	-	-	-
		225	1	477.4	4567	46	18	5	75	0
		250	1	55.68	527	43	7	5	20	0
		1000	2	38.05	335	43	3.5	4.5	18	0
	.67	200	0	-	-	-	-	-	-	-
		225	1	494.62	3632	43	21	5	140	0
		250	2	81.49	675	41	10	5	24.5	0
		1000	3	19.93	190.3	39.7	1.7	3.3	17	0
	1	200	2	155.88	1134.5	43.5	19	5	40	0
		225	1	435.19	4117	42	20	5	83	0
		250	1	15.47	128	39	0	5	8	0
		1000	3	10.67	106.7	36.7	1	3.7	18.7	0
35	.67	225	0	-	-	-	-	-	-	-
		250	0	-	-	-	-	-	-	-
		275	0	-	-	-	-	-	-	-
		1000	3	70.98	433.3	49	6.3	5	25.3	0
	1	225	0	-	-	-	-	-	-	-
		250	1	404.33	1698	54	45	5	96	0
		275	0	-	-	-	-	-	-	-
		1000	3	102.5	705	46	6.3	4.7	22.3	0

TABLE 2

Results for non-Euclidean problems

n	u	L	number of successful problems out of 3	time (seconds)	number of simplex iterations	maximum size of the inverse	maximum level of the search tree	number of Gomory cuts	number of distance constraints	number of capacity constraints	number of subtour prevention constraints
15	0	150	1	315.11	6808	28	28	5	50	276	81
		175	3	109.4	2161	28.3	20	5	6	132.3	22.7
		200	3	145.87	2826.7	29.7	19	5	10.7	156.3	34
		1000	3	63.82	1338	29	16.7	5	0	71.3	11.7
	.33	150	1	15.89	480	24	12	5	2	20	0
		175	3	92.37	1936.3	25.7	24.3	5	16	56	8
		200	3	110.46	2168.7	25.3	24.3	5	11.3	64.7	12
		1000	3	46.34	1071.3	25	15.3	5	0	46.3	3.3
	.67	150	3	.66	38.7	15.7	0	0	0	2.3	0
		175	3	.66	38.7	15.7	0	0	0	2.3	0
		200	3	.66	38.7	15.7	0	0	0	2.3	0
		1000	3	.66	38.7	15.7	0	0	0	2.3	0
	1	150	3	1.37	43	18	0	2.7	4.7	2	0
		175	3	.62	31	16	0	.3	1.3	2	0
		200	3	.45	28	15	0	0	0	1	0
		1000	3	.45	28	15	0	0	0	1	0
20	0	150	1	376.38	4745	37	21	5	4	263	48
		175	2	421.55	5230.5	38	22	5	10	224	57
		200	2	277.64	3564.5	35.5	19.5	5	1	187	36.5
		1000	2	252.15	3175	36.5	17.5	5	0	179	30.5
	.33	150	0	-	-	-	-	-	-	-	-
		175	2	463.28	5584	32.5	35	5	9	155.5	2
		200	2	267.91	2746	31.5	25.5	5	0	103.5	0
		1000	3	330.33	3519	31.3	29.3	5	0	123	0
	.67	150	2	3.72	61.5	25.5	0	2.5	0	6.5	0
		175	2	3.72	61.5	25.5	0	2.5	0	6.5	0
		200	2	3.72	61.5	25.5	0	2.5	0	6.5	0
		1000	3	3.22	63.3	24.7	0	2	0	5	0
	1	150	2	9.58	204	25.5	5.5	3	5	3	0
		175	2	1.68	46	22.5	0	.5	1	2.5	0
		200	3	3.74	69.7	24	1	2	4	4	0
		1000	3	1.48	44.7	22	0	.7	0	2	0
25	0	175	0	-	-	-	-	-	-	-	-
		200	1	247.15	1928	42	26	5	0	130	8
		225	1	235.75	1823	40	31	5	0	120	14
		1000	1	384.54	3014	42	24	5	0	173	8
	.33	175	2	176.17	1231	34	16	3.5	0	57	0
		200	2	176.19	1231	34	16	3.5	0	57	0
		225	2	176.19	1231	34	16	3.5	0	57	0
		1000	2	176.19	1231	34	16	3.5	0	57	0
	.67	175	3	4.96	62.3	28	0	1.7	0	5	0
		200	3	4.96	62.3	28	0	1.7	0	5	0
		225	3	4.96	62.3	28	0	1.7	0	5	0
		1000	3	4.96	62.3	28	0	1.7	0	5	0
	1	175	3	27.93	376	28.7	8.3	2.3	14	6.3	0
		200	3	2.24	48.7	26	0	.7	0	3	0
		225	3	2.24	48.7	26	0	.7	0	3	0
		1000	3	2.24	48.7	26	0	.7	0	3	0

TABLE 2 (continued)

n	ε	L	number of successful problems out of 3	time (seconds)	number of simplex iterations	maximum size of the inverse	maximum level of the search tree	number of Gomory cuts	number of distance constraints	number of capacity constraints	number of subtour prevention constraints
30	0	175	0	-	-	-	-	-	-	-	-
		200	0	-	-	-	-	-	-	-	-
		225	1	383.47	1791	51	20	5	0	139	3
		1000	0	-	-	-	-	-	-	-	-
	.33	175	1	6.36	102	37	0	0	0	10	0
		200	1	6.36	102	37	0	0	0	10	0
		225	1	6.16	102	37	0	0	0	10	0
		1000	1	6.41	102	37	0	0	0	10	0
	.67	175	3	15.18	114.7	37.7	.3	3.3	0	7.3	0
		200	3	15.18	114.7	37.7	.3	3.3	0	7.3	0
		225	3	15.14	114.7	37.3	.3	3.3	0	7.3	0
		1000	3	15.14	114.3	37.3	.3	3.3	0	7.3	0
	1	175	3	29.04	247.7	37.7	4	5	6.7	10.7	0
		200	3	25.07	230	35.3	4	3.3	4.7	8.3	0
		225	3	24.70	227	35.3	3.7	3.3	4	8.3	0
		1000	3	9.41	97.7	33.3	.7	1.7	0	4.7	0
35	.67	175	3	9.97	100.3	37	0	1	0	4.3	0
		200	3	9.97	100.3	37	0	1	0	4.3	0
		225	3	9.97	100.3	37	0	1	0	4.3	0
		1000	3	9.97	100.3	37	0	1	0	4.3	0
	1	175	2	13.35	125	38	0	2	2	5.5	0
		200	3	18.79	114	39	0	2.7	0	5.3	0
		225	3	18.79	114	39	0	2.7	0	5.3	0
		1000	3	18.79	114	39	0	2.7	0	5.3	0
40	.67	175	3	44.52	240.7	43.3	4	1.7	0	10.7	0
		200	3	44.48	240.7	43.3	4	1.7	0	10.7	0
		225	3	44.41	240.7	43.3	4	1.7	0	10.7	0
		1000	3	44.40	240.7	43.3	4	1.7	0	10.7	0
	1	175	2	26.22	154	45	1	4.5	6	2	0
		200	3	21.13	126.7	43.7	.7	3.7	2.7	1.7	0
		225	3	15.89	110.3	42.7	0	3	0	1.3	0
		1000	3	16.28	110.3	42.3	0	3	0	1.3	0
45	.67	175	3	128.58	541	50	3.7	2.7	0	12.3	0
		200	3	128.52	541	50	3.7	2.7	0	12.3	0
		225	3	128.58	541	50	3.7	2.7	0	12.3	0
		1000	3	127.62	541	50	3.7	2.7	0	12.3	0
	1	175	3	161.05	774.7	50	13.7	3.7	6.7	10	0
		200	3	83.86	430.7	47.7	6	2	5.3	6.7	0
		225	3	85.11	430.7	47.7	6	2	5.3	6.7	0
		1000	3	13.83	124.7	45	0	0	0	1.3	0
50	.67	175	3	107.6	342.5	55	3.5	3	0	9.5	0
		200	3	107.6	342.5	55	3.5	3	0	9.5	0
		225	3	107.6	342.5	55	3.5	3	0	9.5	0
		1000	3	107.6	342.5	55	3.5	3	0	9.5	0
	1	175	1	184.28	716	57	17	5	12	8	0
		200	1	67.92	205	56	3	5	8	8	0
		225	2	29.36	155	51	0	1	0	4	0
		1000	3	29.36	155	51	0	1	0	4	0
55	.67	175	3	67.91	219.3	61	0	3.7	0	7	0
		200	3	67.91	219.3	61	0	3.7	0	7	0
		225	3	67.91	219.3	61	0	3.7	0	7	0
		1000	3	67.91	219.3	61	0	3.7	0	7	0
	1	175	2	213.06	773.5	60.5	5.5	5	11	15	0
		200	3	168.69	667.7	58.7	6.7	3.3	2	6	0
		225	3	165.21	660.7	58	6.7	3.3	1.3	5	0
		1000	3	41.44	163.7	57	0	2.7	0	3	0

BIBLIOGRAPHY

[1] BODIN, L.D., GOLDEN, B.L., ASSAD, A., BALL, M.: "Routing and Scheduling of Vehicles and Crews. The State of the Art", Computers & Operations Research, vol.10, pp.69-211, 1983.

[2] CHRISTOFIDES, N.: "The Vehicle Routing Problem", RAIRO (recherche opérationnelle), vol.10, pp.55-70, 1976.

[3] CHRISTOFIDES, N., MINGOZZI, A., TOTH, P.: "The Vehicle Routing Problem", in Combinatorial Optimization, pp.315-338, Wiley, 1979.

[4] CHRISTOFIDES, N., MINGOZZI, A., TOTH, P.: "Exact Algorithms for the Vehicle Routing Problem, Based on Spanning Tree and Shortest Path Relaxations", Mathematical Programming, vol.20, pp.255-282, 1981.

[5] DANTZIG, G.B., FULKERSON, R., JOHNSON, S.M.: "Solution of a Large-Scale Travelling-Salesman Problem", Operations Research, vol.2, pp.393-410, 1954.

[6] GOLDEN, B.L., MAGNANTI, T.L., NGUYEN, H.Q.: "Implementing Vehicle Routing Algorithms", Networks, vol.7, pp.113-148, 1977.

[7] GOMORY, R.: "An Algorithm for Integer Solutions to Linear Programs", Recent Advances in Mathematical Programming, pp.269-302, McGraw-Hill, 1963.

[8] HELD, M., KARP, R.M.: "The Travelling Salesman Problem and Minimum Spanning Trees: Part II", Mathematical Programming, vol.1, pp.6-25, 1971.

[9] LAND, A.H., POWELL, S.: "FORTRAN Codes for Mathematical Programming: Linear, Quadratic and Discrete", Wiley, 249 pp., 1973.

[10] LAPORTE, G.: "Permutation Programming: Problems, Methods and Applications", Ph.D. Thesis, University of London, 358 pp., 1975.

[11] LAPORTE, G., NOBERT, Y.: "A Cutting Planes Algorithm for the m-Salesmen Problem", Journal of the Operational Research Society, vol.31, pp.1017-1023, 1980.

[12] LAPORTE, G., NOBERT, Y.: "A Branch and Bound Algorithm for the Capacitated Vehicle Routing Problem", Operations Research Spektrum, vol.5, pp.77-85, 1983.

[13] LAPORTE, G., NOBERT, Y., DESROCHERS, M.: "Two Exact Algorithms for the Distance Constrained Vehicle Routing Problem", Cahiers du GERAD G-82-05, Ecole des Hautes Etudes Commerciales de Montréal, 19 pp., 1982 (to appear in Networks, 1984).

[14] NOBERT, Y.: "Construction d'algorithmes optimaux pour des extensions au problème du voyageur de commerce", Thèse de doctorat, Département d'Informatique et de Recherche Opérationnelle, Université de Montréal, 242 pp., 1982.

POOR

QUALITY

ORIGINAL